What's new(RMark) R Documentation

Summary of changes by version

Details

Version 1.2

- By default the PIM structure is simplified to use the fewest number of unique parameters. This reduces the size of the design matrix and should reduce run times.
- The above change was made in some versions still numbered 1.1, but it contained an error that caused the links command for MARK to be constructed incorrectly.
- `adjust` argument has been added to `collect.models` to enable control of number of parameters and resulting AIC values.
- `model.list` in `model.table` and `adjust.chat` can now also be a list of models created by `collect.models` which allows operating on sets of models.

Version 1.3

- Time varying covariates can now be included in the model formula. See `make.mark.model` for details.
- New model types for Known (Known-fate) and Multistrata (CJS with different strata) were added. See `Blackduck` and `mstrata` for examples.
- Specific rows of the design data can now be removed for parameters that should not be estimated. Default fixed values can be assigned. The function `make.design.data` now accepts an argument `remove.unused` which can be used to automatically remove unused design data for nested models. It's behavior is also determined by the new argument `default.fixed` in `make.mark.model`.
- `summary.mark` now produces a summary object and `print.summary.mark` prints the summary object. Changes were made to output when `se=T`.
- A new function `merge.occasion.data` was created to add occasion specific covariates to the design data.
- New functions `mark.wrapper` and `create.model.list` were created to automate running models from a set of model specifications for each model parameter.
- The argument `begin.time` in `process.data` can now be a vector to enable a different beginning time for each group.
- An argument `pim.type` was added to parameter specification to enable using pims with time structure for data sets with a single release cohort for CJS. See `make.design.data`
- Model lists created with `collect.models` are now given the class "marklist" which is used with `cleanup` and `print.marklist` (see `print.mark`).
- The function `collect.models` now places the model.table at the end of the returned list such that each model number in model.table is now the element number

in the returned list. Previously it was 1+ that number.
- Input, output, v-c and residual results files from MARK are now stored in the directory containing the `.Rdata` workspace. They are numbered consecutively and the field `output` contains the base filename. The function `cleanup` was created to delete files that are no longer linked to `mark` or `marklist` objects.
- Model averaged estimates and standard errors of real parameters can be obtained with the function `model.average`.

Version 1.4

- Robust design models added. See `robust` for an example.
- Function `cleanup` was modified so warning messages/errors do not occur if no models/files are found.
- Parameters in the design matrix are now ordered in the same consistent arrangement. In prior versions they were arranged based on their order in the argument call.
- Argument `right` was added to `make.design.data`, `add.design.data` and in `design.parameters` of `make.mark.model` to control whether bins are inclusive on the right (default). The `robust` example uses this argument in a call to `mark`.

Version 1.4.1

- A value "constant" was added for the argument `pim.type`. Note that `pim.type` is only used for triangular PIMS. See `make.design.data`
- Some code changes were made to `make.mark.model` which lessen time to create the MARK input file for large models.
- Function add.design.data was modified to accomodate robust design and deletion of design data; this was missed in v1.4 changes.
- `model.name` argument in `mark` and `make.mark.model` was not working. This was fixed.

Version 1.4.2

- Errors in the FORTRAN code were preventing completion of large batch jobs. Now these errors are caught and models that fail are reported and skipped over. Unfortunately, it does require user intervention to close the popup window. Make sure you select Yes to close the window especially if you use the default `invisible=FALSE` such that the window does not appear. If you select No, you will not able to close the window and R will hang.
- A new list element was added to `parameters` in `make.design.data` for parameters such as `Psi` to set the value of tostratum that is computed by subtraction. The default is to compute the probabilitity of remaining in the stratum. The following is an example with strata A to D and setting A to be computed by subtraction for each stratum:

```
ddl=make.design.data(data.processed,
parameters=list(Psi=list(pim.type="constant",subtract.stratum=c
("A","A","A","A")),
p=list(pim.type="constant"),S=list(pim.type="constant")))
```

Version 1.4.3

- Problem with pop up window has been fixed. It will no longer appear if the model does not converge but the model will show as having failed.
- An error was fixed in extracting output from the MARK output file when for some circumstances the label for beta parameters included spaces. This now works properly.

Version 1.4.4

- By including the test on model failure, errors that would stop program were not being displayed. This has been fixed in this version.
- An error was fixed in using time-varying covariates when some of the design data had been deleted.

Version 1.4.5

- For multistrata models, the code for creating the mlogit links for Psi was not working properly if there was more than one group. This was fixed in this version.
- Simplification of the PIMS has now been extended to include mlogit parameters. That was not a trivial exercise and while I feel confident it is correct, double check the assignment of mlogit links for complex models, as I have not checked many examples at present. Within a stratum, the corresponding elements for Psi for each of the tostratum (movement from stratum to each of the other strata excluding the `subtract.stratum`) should have the same mlogit(xxx) value such that it can properly compute the value for `subtract.stratum` by subtraction.

Version 1.4.6

- Assurance code was added to test that the mlogits were properly assigned. An error message will be given if there has been any unforeseen problem created by the simplification. This eliminates any need for the user to check them as described under 1.4.5 above.

Version 1.4.7

- An error was fixed in the `Psi` simplification code. Note that with the fix in 1.4.2 to trap errors, a side effect is that non-trapped errors that occur in the R code will now fail without any error messages. If the error occurs in making the model, then the

model will not be run, but you will not receive a message that the model failed. I may have to make the error trapping a user-settable option to provide better error tracking.

Version 1.4.8

- Argument `silent` was added to `mark` and `mark.wrapper` with a default value of `FALSE`. This overcomes the problem described above in 1.4.7.
- Code was added to `collect.model.names` to prevent it from tripping up when files contain an asterisk which R uses for special names.
- Use of T and F was properly changed to TRUE and FALSE in various functions to prevent errors when T or F are R objects.
- Code for naming files was modified to avoid problems when more than 999 analyses were run in the same directory.
- Bug in setting fixed parameters with argument `fixed=list(index=,value=)` was corrected.
- Argument `remove.intercept` was added to parameter definition to force removal of intercept in designs with nested factor interactions with additional factor variables (e.g., `Psi=list (formula=~sex+stratum:tostratum,remove.intercept=TRUE)`).

Version 1.4.9

- Argument `initial` of `make.mark.model` was not working after model simplification was added in v1.2. This was modified to select initial values from the model based on names of design matrix columns rather than column contents which have different numbers of rows depending on the simplification.
- `extract.mark.output` was fixed to extract the correct -2LnL from the output file in situations in which initial values were specified.

Version 1.5.0

- If output file already exists user is given option to create mark model from existing files. Only really useful if a bug occurs (which occurred to me from 1.4.9 changes) and once fixed any models already run can be brought into R by running the same model over and specifying the existing base `filename`. Base `filename` values are no longer prefixed with MRK to enable this change.
- On occasion MARK will complete the analysis but fail to create the v-c matrix and v-c file. The code has been modified to skip over the file if it is missing and output a warning.
- Two new functions have been added to ease handling of marklist objects. `merge.mark` merges an unspecified number of marklist and mark model objects into a new marklist with an optional model.table. `remove.mark` can be used to remove mark models from a marklist. See `dipper` for examples of each function.

- Various changes were made to functions that compute real parameter estimates, their standard errors, confidence intervals and variance-covariance matrix. The functions that were changed include `compute.real`,`find.covariates`,`get.real`,`fill.covariates`. For examples, see help for latter two functions.

Version 1.5.1

- Functions `compute.link` and `get.link` were added to compute link values rather than the parameter estimates.
- A function `convert.link.to.real` was added to convert link estimates to real parameter estimates. Previously a similar internal function was used within `compute.real` but to provide more flexibility it was put into a separate function.
- An argument `beta` was added to `get.real` to enable it to be changed in the computation of the real parameters rather than always using the values in `results$beta`.
- A function `TransitionMatrix` was added to create a transition matrix for the Psi values. It is provided for all strata including the `subtract.stratum`. Standard errors and confidence intervals can also be returned.
- `make.mark.model` was modified to include `time.intervals` as an element in the mark object.

Version 1.5.2

- Two new functions were added in this version. `convert.inp` converts a MARK encounter history input file to an RMark dataframe. This will be particularly useful for those folks who have already been using MARK. Instead of converting and importing their data with `import.chdata` they can use the `convert.inp` to import their .inp file directly. It can also be used to directly import any of the example .inp files that accompany MARK and the MARK electronic book (http://www.phidot.org/software/mark/docs/book/). The second new function is only useful for tutorials and for first time users trying to understand the way RMark works. The function `PIMS` displays the full PIM structure or the simplified PIM structure for a parameter in a model. The user does not directly manipulate PIMS in RMark and they are essentially transparent to the user but for those with MARK experience being able to look at the PIMS may help with the transition.

Version 1.5.3

- In function `get.real` a fix was made to accommodate constant pims and a warning is given if the v-c matrix for the betas has non-positive variances.
- In function `make.mark.model`, the argument `initial` can now be a single value which is then assigned as the initial value for all betas. I have found this useful for POPAN models. For some models I have run, the models fail to converge in MARK

with the default initial values it uses (I believe it uses `initial=0`). I have had better luck using `initial=1`. By allowing the use of a single value you can use the same generic starting value for each model without figuring out the number of betas in each model. Also note that you can specify another model that has already been run to use as initial values for a new model and it will match parameter values.

- A bad bug was fixed in `cleanup` which was unfortunately deleting files containing "out", "inp", "res" or "vcv" rather than those having these as extensions. This happened without your knowledge if you chose ask=FALSE. Good thing I had a backup. Anyhow, I have now restricted it to files that are named by RMark with markxxxx.inp etc where xxxx is a numeric value. Thus if you assign your own basefile name for output files you'll have to delete them manually. Better safe than sorry.

Version 1.5.4

- In function `mark` an argument `retry` was added to enable the analysis to be re-run up to the number of times specified. An analysis is only re-run if there are "singular" beta parameters which means that they are either non-estimable (confounded) or they are at a boundary. Beginning with this version, `extract.mark.output` was modified such that the singular parameters identified by MARK are extracted from the output (if any) and the indices for the beta parameters are stored in the list element `model$results$singular`. The default value for `retry` is 0 which means it will not retry. When the model is re-run the initial values are set to the values at the completion of the last run except for the "singular" parameters which are set to 0. Using `retry` will not help if the parameters are non-estimable. However, if the parameters are at a boundary because the optimization "converged" to a sub-optimal set of parameters, then setting `retry` to 1 or a suitably small value will often help it find the MLEs by moving away from the boundary. If the parameters are estimable and setting `retry` does not work, then it may be better to set new initial parameters by either specifying their values or using a model with similar parameters that did converge.
- A new function `rerun.mark` was created to simplify the process of refitting models with new starting values when the models were initially created with `mark.wrapper` which runs a list of models by using all combinations of the formulas defined for the various parameters in the model. Thus, individual calls to `mark` are not constructed by the user and re-running an analysis from the resulting list would require constructing those calls. The argument `model.parameters` is now stored in the model object and it is used by this new function to avoid constructing calls to rerun the analysis. With this new function you only need to specify the model list element to be refitted, the processed dataframe, the design data and the model list element (or different model) to be used for initial values. See `rerun.mark` for an example.
- To make `rerun.mark` a viable approach for all circumstances, the functions

mark.wrapper and model.table were modified such that models that fail to converge at the outset (i.e., does not provide estimates in the output file) are stored in the model list created by the former function and they are reported as models that did not run and are skipped in the model.table by the latter function. This enables a failed model to be reanalyzed with rerun.mark using another model that converged for starting values.

Version 1.5.5

- model has been deleted from the arguments in TransitionMatrix. It was only being used to ascertain whether the model was a Multistrata model. This is now determined more accurately by looking for the presence of tostratum in the argument x which is a dataframe created for Psi from the function get.real. The function also works with the estimates dataframe generated from model.average. See help for TransitionMatrix for an example.
- An argument vcv was added to function model.average. If the argument is TRUE (the default value) then the var-cov matrix of the model averaged real parameters is computed and returned and the confidence intervals for the model averaged parameters are constructed. Models with non-positive variances for betas are reported and dropped from model averaging and the weights are renormalized for the remaining models.
- A new function compute.links.from.reals was added to the library to transform real parameters to its link space. It has 2 functions both related to model averaged estimates. Firstly, it is used to transform model averaged estimates so the normal confidence interval can be constructed on the link values and then back-transformed to real space. The second function is to enable parametric bootstrapping in which the error distbution is assumed to be multivariate normal for the link values. From a single model, the link values are easily constructed from the betas and design matrix so this function is not needed. But for model averaging there is no equivalent because the real parameters are averaged over a variety of models with the same real parameter structure but differing design structures. This function allows for link values and their var-cov matrix to be created from the model averaged real estimates.

Version 1.5.6

- print.summary.mark was modified so fixed parameters are noted.
- Argument show.fixed was added to summary.mark to control whether fixed parameters are shown as NA (FALSE) or as the value at which they were fixed. If se=T the default is show.fixed=T otherwise show.fixed=F. The latter is most useful in displaying values in PIM format (without std errors), so fixed values are displayed as blanks instead of NA.
- Argument links was added to convert.link.to.real and the default value for

argument `model` is now NULL. One or the other must be given. If the value for `links` is given then they are used in place of the links specified in the `model` object. This provides for additional flexibility in changing link values for computation (eg use of log with mlogit).
- Argument `drop` was added to `model.average`. If `drop=TRUE` (the default), then any model with one or more non-positive (0 or negative) variances is not used in the model averaging.
- An error in computation of the v-c matrix of mlogit link values in `compute.links.from.reals` was fixed. This did not affect confidence intervals for real parameters (eg Psi) in `model.average` because it uses the logit transformation for confidence intervals on real parameters that use mlogit link (eg Psi).
- `get.real` was unable to extract a single parameter value(eg constant Phi model). This was fixed.
- The argument `parm.indices` was removed from the functions `compute.real` and `convert.link.to.real` because the subsetting can be done easily with the complete results returned by the functions. This changed the examples in `fill.covariates`.
- `compute.real` and subsequently `get.real` return a field `fixed` when se=T that denotes whether a real parameter is a fixed parameter or an estimated parameter at a boundary which is identified by having a standard error=0.

Version 1.5.7

- Argument `data` was added to function `model.average` to enable model averaging parameters at specific covariate values rather than the mean value of the observed data. An example is given in the help file.
- Argument `parameter` of function `model.average` now has a default of NULL and if it is not specified then all of the real parameters are model averaged rather than those for a particular type of parameter (eg p or Psi).
- A bug was fixed in function `compute.real` that caused the function to fail for computations of `Psi`.

Version 1.5.8

- Argument `options` was added to `mark` and `make.mark.model` with a default NULL value. It is simply a character string that is tacked onto the `Proc Estimate` statement for the MARK .inp file. It can be used to request options such as NoStandDM (to not standardize the design matrix) or SIMANNEAL (to request use of the simulated annealing optimization method) or any existing or new options that can be set on the estimate proc.
- A bug in `model.table` was fixed so it would accomodate the change from v1.3 to a marklist in which the model.table was switched to the last entry in the list.

- A bug in summary.mark was fixed so it would properly display QAICc when chat > 1.
- Function adjust.chat was modified such that it returns a marklist with each model having a new chat value and the model.table is adjusted for the new chat value.
- Function adjust.parameter.count was modfied so it returns the mark model object rather than using eval to modify the object in place. The latter does not work with models in a marklist and calls made within functions.

Author(s)

Jeff Laake

[Package *RMark* version 1.5.7 Index]